71-01578

JOHN F. KENNEDY SPACE CENTER

KSC-TR-1111

MAY 18 1971

N72-14216 (NASA-TM-X-67567) REQUIREMENT FOR A STANDARD LANGUAGE FOR TEST AND GROUND OPERATIONS J.R. Medlock (NASA) 17 May 1971 26 p

Unclas 11872

CSCL 09B G3/08

TECHNICAL REPORT

REQUIREMENTS FOR A STANDARD LANGUAGE FOR TEST AND GROUND OPERATIONS

DIRECTORATE OF LAUNCH OPERATIONS

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
KSC-TR-1111		

4. Title and Subtitle	5. Report Date
Requirements For A Standard Language For Test And Ground Operations	May 17, 1971
	6. Performing Organization Code: LV-CAP

7. Author(s)	8. Performing Organization Report No.
Joe R. Medlock	

9. Performing Organization Name and Address	10. Work Unit No.
John F. Kennedy Space Center Kennedy Space Center, Florida 32899	11. Contract or Grant No.
	13. Type of Report and Period Covered: TECHNICAL REPORT
12. Sponsoring Agency Name and Address	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

This report contains the basic requirements for a standard test and checkout language applicable to all phases of the Space Shuttle test and ground operations. While the concentration of this effort has been toward the Space Shuttle needs, the language should be sufficiently viable for use in subsequent Space Station test operations. The general characteristics outlined herein represent the integration of selected ideas and concepts from operational elements within Kennedy Space Center (KSC) that represent diverse disciplines associated with space vehicle testing and launching operations. Special reference is made to two studies conducted in this area for KSC as authorized by the Advanced Development Element of the Office of Manned Space Flight (MSF). Information contained in reports from these studies have contributed significantly to the final selection of language features depicted in this technical report. The contractors that performed the studies were Martin Marietta, Denver, Colorado, and M & S Computing, Inc., Huntsville, Alabama. Their final reports are listed as references 1 and 2, respectively.

17. KeyWords	18. Distribution Statement
Computer Language Standard Test Language Space Shuttle Test Language	

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified		N/A

PRECEDING PAGE BLANK NOT FILMED

TABLE OF CONTENTS

ACRONYMS/ABBREVIATIONS

ACE-S/C	- Acceptance Checkout Equipment for Spacecraft
ASCII	- USA Standard Code for Information Interchange
C/D RFQ	- Phase C/D Shuttle RFQ
DIU	- Digital Interface Unit
EBC DIC	- Extended Binary Coded Decimal Interchange Code
EBW	- Exploding Bridge Wire
ECLS	- Environmental Control and Life Support
GSE	- Ground Support Equipment
KSC	- Kennedy Space Center
LRU	- Line Replaceable Unit
MSFC	- Marshall Space Flight Center
MSC	- Manned Spacecraft Center
PU	- Propellant Utilization
RF	- Radio Frequency
RFQ	- Request for Quote
SATURN V IP & CL	- Saturn V Instrumentation Parts & Components List
TM	- Telemetry
VAB	- Vehicle Assembly Building

REQUIREMENTS FOR A STANDARD LANGUAGE FOR TEST AND GROUND OPERATIONS

SECTION I

INTRODUCTION

1.1 BACKGROUND

The initiative to produce a standard test language f the Space Shuttle emerged from experiences gained in test automation during the Saturn/ Apollo prog.am. The standardized test language concept was introduced to the launch site by the Marshall Space Flight Center (MSFC) during the development phase of the ground support system for the Uprated Saturn I program. This concept has grown to a place of prominence in the testing and launching of Apollo space vehicles.

During early Saturn checkout operations the typical user-programmer communication gap was experienced firsthand as test personnel attempted to comprehend the programmer's interpretation of their test requirements. Generally, changes were difficult to implement and to understand. Full control over the test came only after much actual experience. Even then the details of some operations were obscurely embedded in the test packages so that the test engineers had difficulties in comprehending the subtleties of the programmer's logic. At the time when automatic checkout could have eased the mounting strain associated with the pressing schedule, the lack of a common language to communicate requirements and to describe the computer programs further burdened the launch team. Major problems arose from the lack of concise uniform test notations that could be readily understood by all the different engineering elements and were sufficiently definitive enough to serve as a detailed test requirement for automatic program generation.

It was through the insistence and persistence of a group of NASA personnel that a basic set of coded operations for accomplishing the minimum engineering functions was added to the Uprated Saturn I Ground Computer Operating System. During the early application, test personnel learned that the success of the computer language could be strongly dependent upon its efficient implementation through the language processor and its execution through the operating system (real time executive). The first attempt at implementing automatic test procedures in the language was operationally inefficient, largely because of limitations in these two areas.

For Saturn/Apollo we must place ourselves in the same category for language development as most of the other groups who develop test languages within industry or Government. That is, the language was developed only after the equipment and applications were firmly established and was among the last items to be implemented. Under this circumstance, the language was destined to become the top contender for criticism as another new, unfamiliar, troublesome system black box that complicated the engineer's life and added to his growing list of problems.

The language has evolved to the point where most vehicle disciplines use the language extensively to prepare automatic test procedures. It is from this practical experience that final selection of language features was made.

We are now presented with the rare opportunity of standardizing the basic communications among all facets of ground and in-flight testing at a point in the system acquisition cycle where it will become a natural part of the program. This will avoid costly retrofits and difficult 'unlearning' exercises at a later time. The language will serve as the basic tool in insuring commonality for Orbiter/Booster/GSE Test and Ground Operations Procedures while inherently providing the capability to: (1) efficiently automate manual procedures, (2) readily adapt design procedures for operational use, (3) reduce supporting documentation, (4) efficiently cross-train test personnel, (5) minimize impacts from changes, and (6), in general, will be a prime contributor in support of the rapid turnaround requirements.

Cost effectiveness considerations which justify a standard test language include the following:

(1) Economy realized in programming manhours for the high order language over the machine symbolic language. The ratio in favor of the high order language is approximately ten to one for an equivalent job.

(2) Program turnaround time significantly reduced for programs written in the language.

(3) Documentation costs significantly reduced because the language provides its own documentation.

(4) The language allows programs to be written and maintained by the responsible system engineers, thereby reducing the need for a programming group dedicated for this purpose.

(5) Computer programs more readily and efficiently controlled and maintained at the high order language level.

The necessity for a standard test language in this environment cannot be overemphasized; however, care should be exercised in selecting the scope of tasks that a language describes. The assertion that 'one language should be used for everything' sounds attractive, but under close examination this approach would defeat the objective for simplicity and readability. If a single universal language were defined for the real-time operating systems, off-line compilers, data bus communication, guidance and navigation, as well as test procedures, the requirements would differ so widely that the resulting composite would be a high-level machine language extremely difficult for personnel outside of the computer field to use and understand. Test (Ground/Inflight) and ground operations procedures represent a logical subdivision of the total task, and the language supporting these areas should be capable of defining most of the required activities. While preserving the general readability such a capability would help minimize the tedious, costly, time-consuming traditional interface between the test engineer and the programmer.

For the Space Shuttle Program, we must make maximum use of the lessons learned through the years of design and launch experience. The very nature of the Space Shuttle design and the essence of the operational concept dictate that more be accomplished in a shorter period by fewer people than ever before. Automation, then, becomes a requirement for operations, not an elective. To effectively apply extensive automation, a test language has no suitable alternate.

1.2 PURPOSE

To define a standardized test language applicable for all phases of Space Shuttle testing and for ground operations.

1.3 SCOPE

The requirements for a comprehensive test and checkout language includes the capability for specifying procedures required for the following:

(1) LRU Bench Testing

(2) Single System Testing

(3) Concurrent Systems Testing

(4) Vehicle End-to-End Testing

(5) Fault Isolation and Diagnostic Testing

(6) Overall Maintenance Testing

(7) Inflight Testing

(8) LRU and System Certification

(9) Payload Test and Preparation

(10) Launch Preparation

(11) Propellant Loading

(12) Vehicle Safing

(13) Vehicle Monitoring

(14) Backout and Contingency Operations

(15) Training and Simulation

1.4 APPROACH

The background information was obtained by a survey of test-oriented languages that had been designed for various test applications on Government and industry projects. The current proposals for the Space Shuttle were then analyzed and language characteristics peculiar to the Space Shuttle were developed. From these two efforts, a complete list of language requirements, consistent with the design concepts of the Space Shuttle, was produced. The above was accomplished by KSC Contracts with Martin Marietta and M & S Computing. At the time the contracts were awarded, a Technical Team was organized to insure representation from a broad technical base. The membership was selected from key personnel involved in space vehicle design, assembly, verification, launch, and support from KSC, MSC, and MSFC. This report is the initial output from this team and is intended to be maintained as a working document. Changes will be made, as required, to assure compatibility with the developing Space Shuttle design.

SECTION II

REQUIREMENTS

2.1 GENERAL

This section presents a summary of Shuttle Test Requirements, many of which will be automated. The Language Requirements, the substance of this report, which are detailed in paragraph 2.3., are intended to satisfy the needs for testing of the Space Shuttle during ground and inflight operations. The degree to which this language is used inflight must be resolved after the Space Shuttle design becomes more complete.

2.2 TEST REQUIREMENTS

It is quite clear that the Space Shuttle design stresses onboard autonomy to allow operations to resemble airlines practices as closely as possible. Also,the design visualizes redundancy of components and systems to assure mission success in the severe environment of space operations. While these and other Space Shuttle design considerations, in some respects, tend to veer away from the present space vehicle checkout and launch practices, they do not invalidate the need for a common test language, but substantiates this need as more urgent for the Space Shuttle than for Apollo/Saturn.

Paragraphs 2.2.1 through 2.2.5 contain a tabulation of proposed Space Shuttle tests broken down by work areas. This is a tentative tabulation that does not include bench test activities. A survey of the list reveals that approximately 90% of the tabulations are candidates for automation or partial automation.

2.2.1 Safing and Towing (Booster/Orbiter)

a. Apply ground power for safety systems status check.

b. Safety system status check. Verify that ordnance has fired, fuel residuals are zero, and that systems exposed to abnormal heating have no faults (i.e., hydraulic leakage, shorted cables, etc.)

c. During towing, record reaction of landing gear to verify that no degradation of function occurred during the last landing.

2.2.2 Maintenance Area (Booster/Orbiter)

a. Application of diagnostics to confirm inflight LRU switchout.

b. Schedule LRU replacement. Consideration should be given to the following:

(1) Previous History

(2) Interaction with other LUR's

(3) LRU Criticality

(4) Man Loading

(5) Task Time

(6) Retest Requirements

(7) Pending Modifications

(8) Special Tooling Availability

(9) Re-order of LRU's

(10) Work Order Generation

(11) Task Closeout

c. Maintenance Tasks

(1) Generate a maintenance schedule based upon operational readiness requirements and LRU replacements.

(2) Perform steering tests of aerodynamic surfaces.

(3) Perform nosewheel steering tests.

(4) Perform air breather functional test.

(5) Perform tank pressure decay tests.

(6) Verify redundancy.

(7) LRU modification verification.

(8) Perform computer/DIU interface check.

(9) RF and TM checks.

(10) Power on checks.

(11) Monitor system performance to specifications more stringent than those employed in flight.

(12) Load software changes.

(13) Verify Payload interface.

(14) Applications/experiments verification.

(15) Perform ECLS functional.

(16) Verify Booster/Orbiter interface.

(17) Flight critical LRU Tests:

- (a) Engine actuators.
- (b) PU probes.
- (c) Engine timing.
- (d) Vent valves.
- (e) Fill and drain valves.
- (f) Engine data recorders.
- (g) Generators.
- (h) Power transfer devices.
- (i) EBW charging units.
- (j) Fuel cells.

(18) Orbiter engine bell extension.

(19) Avionics operations test.

(20) Data Review.

2.2.3 Mobile Launcher Checkout

a. Systems monitor.

b. Arm and holddown functionals.

c. Mechanical redundancy.

d. Water system functional.

e. Ground Computer GSE Data Bus Interface verification.

f. Modifications checkout.

g. Electronic redundancy verification.

h. Fluid pressure integrity.

i. Propellant simulated load.

2.2.4. VAB and Pad Operations

a. Systems monitor.

b. Hook-up umbilicals.

c. Leak test umbilicals.

d. Orbiter/Booster Power On.

e. Vehicle GSE Interface Test.

f. Flight Electrical Mate and Interface Test.

g. Avionics Operations Test.

h. Data reviews.

i. Countdown preparations.

j. Payload/applications [illegible].

k. Countdown.

2.2.5 Inflight Operations

a. Checkout.

b. Diagnostics.

c. Redundancy management.

d. Consumables management.

e. Sequencing.

f. Payload/experiments operations.

2.3 LANGUAGE REQUIREMENTS

Contained within Language Requirements are the Primary Language Objectives which are general statements of the philosophy to be used in implementing the language and the Primary Language Requirements which focus on the specific actions and performance required of the language.

2.3.1 Primary Language Objectives

a. The language requirements must be consistent with and support the design concepts and requirements of the Space Shuttle.

b. The language will not be constrained by specific test equipment.

c. The language will allow the same procedure to be used for both manual and automatic testing.

d. The language shall provide for a flexible monitoring capability.

e. The language will provide the capability for test personnel to communicate with mission software.

f. The language will be easy to use by test-oriented personnel not necessarily skilled in programming techniques.

g. The language will be easy to read and will be self-documenting.

h. The language must be compatible with the philosophy of performing concurrent testing.

2.3.2 Primary Language Requirements

a. English-like words, structure, and punctuation.

b. Comments.

c. Control of the system under test.

d. Data sampling from the system under test.

e. Data comparison.

f. Time controlled events.

g. Performance monitoring.

h. Information presentation and recording.

i. Console interaction.

j. Data manipulation.

k. Computer-to-computer communication.

l. Interface with other languages.

m. Test sequence designation.

n. Language redundancy.

o. Identification of language packages and components.

p. Data dictionary.

q. Table definition.

r. Writer aids.

s. Reaction to system changes.

t. Language character set.

2.3.2.1 English-like Words, Structure, and Punctuation

The key words of the test language form the building blocks of the language and care should be taken to select words natural to the Space

Shuttle test-environment. Abbreviations should generally be avoided and only in cases where the abbreviation has gained universal acceptance will a deviation be considered. These key words will be ordered in a logical english form. This ordering will promote learning and retention while allowing comprehensive error checking. The punctuation characters and their meaning should be consistent with general usage.

2.3.2.2 Comments

A comment is an expression which clarifies a particular statement or functional aspect of a group of statements but is not required to technically define the procedural actions of the operation. When automated, comments have no effect on the operation of the computer performing the assigned tasks. The capability to easily insert comments is important in that in some applications it provides the added flexibility required to allow the computer listing to be the single control document.

2.3.2.3 Total Control of the System Under Test

The language should allow test personnel to specify test point control to the lowest level that is available under the given system configuration. The level of access to a Line Replaceable Unit will probably be different in the off-line test environment than in the operational system configurations. The types of signals used in controlling the test or function operations must not be constrained by the standardized test language. For the Space Shuttle System, the language must recognize the requirement for discrete events, digital codes, and proportional values (digital representation of analog values).

2.3.2.4 Data Sampling

The test language should support data gathering consistent with system constraints and ground rules. The ability to exercise control over the system under test and the ability to measure parameters and status of the test item are the foundations for testing. Sampling rates should be by a system limit and not a language limit. It is possible, after the system constraints have been defined, to incorporate the constraints into a language processor which will alert the user if his procedure conflicts with system constraints. As with control, data samples may be discrete events (ON/OFF), digital codes (LRU address), or proportional values (0-100%).

2.3.2.5 Data Comparison

Data comparison is the next basic level above the ability to control the test item and the ability to acquire the data from the test article. The

comparison may take many forms (test versus predicted, per cent change from last value, deviation during time interval) and the results may be saved for use later or may immediately effect a change in the processing sequence. The data comparison capability should include arithmetic and Boolean terms in a form familiar to the test environment.

2.3.2.6 Time Controlled Events

The extensive use of time factors in sequencing and testing is a salient feature of test and ground operation procedures. During launch preparations, test and functional operations are often controlled by a specific time relative to liftoff (count down clock). Other functions are based on given time of day; eg., usually referenced to Greenwich Mean Time. Mission elapse time may be used for inflight test and operations. Many of the system sequences must be performed in a close time-controlled sequence relative to an occurrence of an event within the vehicle. There are also those indicators that must be checked at a periodic rate. The comprehensive use of time in testing warrants major consideration in selecting the proper key words to be used in a test language.

2.3.2.7 Monitoring the System Under Test

Recognizing that detailed checkout philosophies have not been defined for future vehicles, an increased dependence upon monitoring is an established trend in the space and airlines industries. Though most of the existing space system checkout facilities include monitoring capabilities, most of the automated test languages seem to exclude this capability. Realizing that interaction with the real time hardware/software system could dictate some adjustments to a predefined test language, the basic language should be able to define items to be monitored; eg., conditioned by time (start/stop and sampling interval). The general capabilities of the language should also be available for specifying monitoring packages.

2.3.2.8 Information Presentation and Recording

In the automation of test requirements, the manner in which the data is presented to the test evaluator can signifantly influence the effort required in deducing the proper action to be taken or determining whether or not all aspects of the test were completed satisfactorily. The ability to record or save selected data, usually correlated with time, is also an operation that is frequently performed throughout system testing and must be supported by the test language.

2.3.2.9 Console Interaction

At times during a test, an anomaly may appear that justifies suspending activity until the system status and integrity can be confirmed. The

decision may be just to resume operating steps, or rerun certain steps, or to deviate in some way by changing test parameters. The basic language requirements to be derived from this situation are:(a) the language must be able to suspend execution until requested to continue, and (b) the language must accommodate the need to change test parameters from a console for certain predefined parameters. This feature is also dependent upon the operational hardware/software system and refinements to the language may result from later system definition.

2.3.2.10 Data Manipulation

Data manipulation is considered to encompas numeric formulas, relational formulas, and computer associated assignment statements. The studies (References 1 and 2) reflect that 'generally, the languages that provide arithmetic capabilites provide these capabilities in a formula type statement (e.g., FORTRAN type statement). This is a reasonably natural and compact way to describe the required calculations (Reference 2).' The relational formulas are of the comparison type usually expressed in a form of 'equal to' or the 'not equal to'. For automatic testing, a closely related requirement exists, which is the moving of data items between storage cells.

2.3.2.11 Computer-to-Computer Communications

It appears that the Space Shuttle will have inter-computer communication in some form. It could be between the central computers, between a central computer and an engine computer, or between a central computer and an off-vehicle computer (ground system or space station system). The test language will provide this capability in a manner that will ensure two-way communication between digital devices. This will then include the capability to transmit and receive data from some of the more complex data bus interface units.

2.3.2.12 Incorporation of Packages Written in Other Languages

'The need to specify certain functions in assembly language is not expected to disappear entirely. It complicates a language considerably to include every capability necessary to handle highly exceptional requirements. However, to ensure that exceptional requirements can be fulfilled, it is necessary to have some capability to incorporate assembly language programming consistent with the design intent of the language (Reference 4).' This feature will probably be used only by the sophisticated test programmer and under a higher level of control and validation than required for packages written in the standard test language. This requirement recognizes that other languages, assembly level or high order, may be needed and the standard test language must support this concept.

2.3.2.13 Test Sequence Designation

A desired test sequence may be stated in several ways. A test procedure usually contains many functional elements. These functional elements are comprised of a varying number of individual operating steps (statements). If the functional element must be repeated a number of times, then it may become a subprocedure and referenced by the main procedure, or the steps of the elements may be inserted the proper number of times, or direction may be given to repeat the required steps the appropriate number of times. This looping type capability is even more important when the procedure is automated because it often has a direct impact on storage allocations required for the procedure. This requirement includes the need for directing the sequence based on the condition of test indicators.

2.3.2.14 Language Redundancy

The selection of the words of the language should reflect the consideration for redundancy. This may result in each word differing from every other word by at least two characters, thus preventing a single mistake from yielding a valid, yet different, meaning. It may be that the best approach would be to provide additional cross checks within the total statement. To the maximum extent practical, pure numerical elements should be avoided. Incorrect numbers are difficult to detect without a sophisticated verification capability.

2.3.2.15 Identification of Language Packages and Components

This includes the obvious need of the ability to reference a specific test procedure for use during testing, for incorporating changes, and for configuration control. Often procedures must reference other procedures. Individual statements within the procedure have the same need; therefore, the language will provide for labeling of separate packages as well as individual statements.

2.3.2.16 Data Dictionary Requirement

The data dictionary concept is the feature of the language that allows the language to be independent of the test equipment. It is basically a cross reference table that relates the engineering terminology of a test point to the test equipment parameters required to access the test point. For example: the procedure might read apply booster measuring power. The dictionary would take 'booster measuring power' and provide the necessary data for the automatic test equipment such as data bus number, interface unit address, line replaceable unit designation, and other system related values necessary to locate the test point and to accomplish the desired results. For the

Space Shuttle, there will probably be a centrally defined and controlled list of test points similar to Apollo documents; eg., the Saturn V Discrete Running List, Saturn V IP&CL, and ACE-S/C Programming Requirements Process Specification Parameter List. Such a document for the Space Shuttle would furnish much of the information required in the dictionary. This is the final link between the language and the test system. It also allows procedures to be written independent of the test system and in advance of the final configuration.

2.3.2.17 Table Definition

Special attention should be given to the definition and use of tables. They should prove to be a significant aid in test preparation. The flexibility and usefulness of table operations warrants the inclusion of this capability even though it might appear more complex than desired. It is expected that most table declarations will be made by more experienced programming oriented personnel and that such declarations will be included in the data dictionary. Tables are currently being used in Saturn checkout procedures and have become an integral part of daily operations and major tests. Generally, they support such functions as system status checks, switch scans, and performance monitoring.

2.3.2.18 Data Types (Reference 3).

Investigation of the Space Shuttle test and checkout applications and previous efforts at the definition of test languages leads to the conclusion that the following constant and data types are required in the new language.

(1) Constants

- (a) Integer
- (b) Fixed Point
- (c) Boolean
- (d) Text
- (e) Binary *
- (f) Time

NOTE: Binary may be expressed as octal or hexadecimal.

(2) Data Variables

(a) Integer

(b) Fixed Point

(c) Boolean

(d) Text

(e) Time

2.3.2.19 Writer Aids

It appears consistent throughout aviation and space vehicle checkout procedures that the writing task is a relatively small portion of the overall procedure cycle. The number of people using, reading, validating, or changing procedures can sometimes become rather large. Therefore, while primary consideration must be given to the larger group, the test procedures writer is certainly a vital link in the cycle and should be afforded the capability required to insure maximum economy without compromising the primary language objectives. The requirement includes such standard concepts as replacing the name of one item with another, macro features, and subrouting capabilities. Portions of other language requirements also may be implemented in a manner which facilitates procedure writing. The final selection must be constrained by the fact that 'the user is ill-served by a language which allows him to conveniently describe an erroneous procedure,... (Reference 5).'

2.3.2.20 Reaction to System Changes

The requirements include the basic needs as being able to respond to such general system indicators as 'start processing,' 'terminate processing,' and 'suspend processing.' The command could have been originated by a manual entry, another procedure, or an internally-generated command due to detection of a serious anomaly. Although the Space Shuttle does not appear to be using system interrupts, the language should be able to accommodate interrupts for component type testing and to preclude a language impact if the Space Shuttle or Space Station implements interrupts at a later time.

2.3.2.21 Language Character Set

To promote general applicability of the language to as many test applications and test equipment as practical, only characters may be used that

are common to the USA Standard Code for Information Interchange Code (ASCII) and the Extended Binary Coded Decimal Interchange Code (EBC DIC). These characters (Reference 3) are as follows:

(1)	Capital Letters:	A-Z		
(2)	Numbers:	0-9		
(3)	Special Characters:	+	:	@
		-	!	blank
		=	?	-
		"	<	*
		'	>	#
		.	(	$
		,	)	%
		;	/	&

SECTION III

IMPLEMENTATION

3.1 GENERAL

The Language Development Schedule shown in Figure 1, indicates the significant language development activities and relates these to the Space Shuttle schedule. The Language Development Activities are briefly described in the following paragraphs.

3.2 TEST LANGUAGE REQUIREMENTS

This document contains the Test Language Requirements and will be revised as deficiencies are identified.

3.3 TEST LANGUAGE SPECIFICATION

The Test Language Specification shall include a complete set of syntactic diagrams with the attendant semantics required to satisfy the language requirements and objectives. Distinction will be made between language characteristics and language processor, operating system, and system interpreter characteristics.

3.4 DATA DICTIONARY CONCEPT

Past experience and analysis have revealed the need for implementation of a Data Dictionary to supply certain declarations, translations from english notation to address patterns, macros, calibration data, and other modules of common usage requiring centralized control. This concept is vital to make the language independent of the test equipment.

While the initial work can be started prior to the C/D contract date, the complete definition of the dictionary concept will be possible only after the avionics contractor has developed the necessary detailed information during the C/D contract. It should be noted that the Test Language Specification can be released and be used for test procedure specification independently of this work.

3.5 SOFTWARE SYSTEM INTERFACE DEFINITION

Tradeoffs must be performed to establish the interface between tasks to be executed by the onboard software components, such as the operating

TEST LANGUAGE DEVELOPMENT SCHEDULE

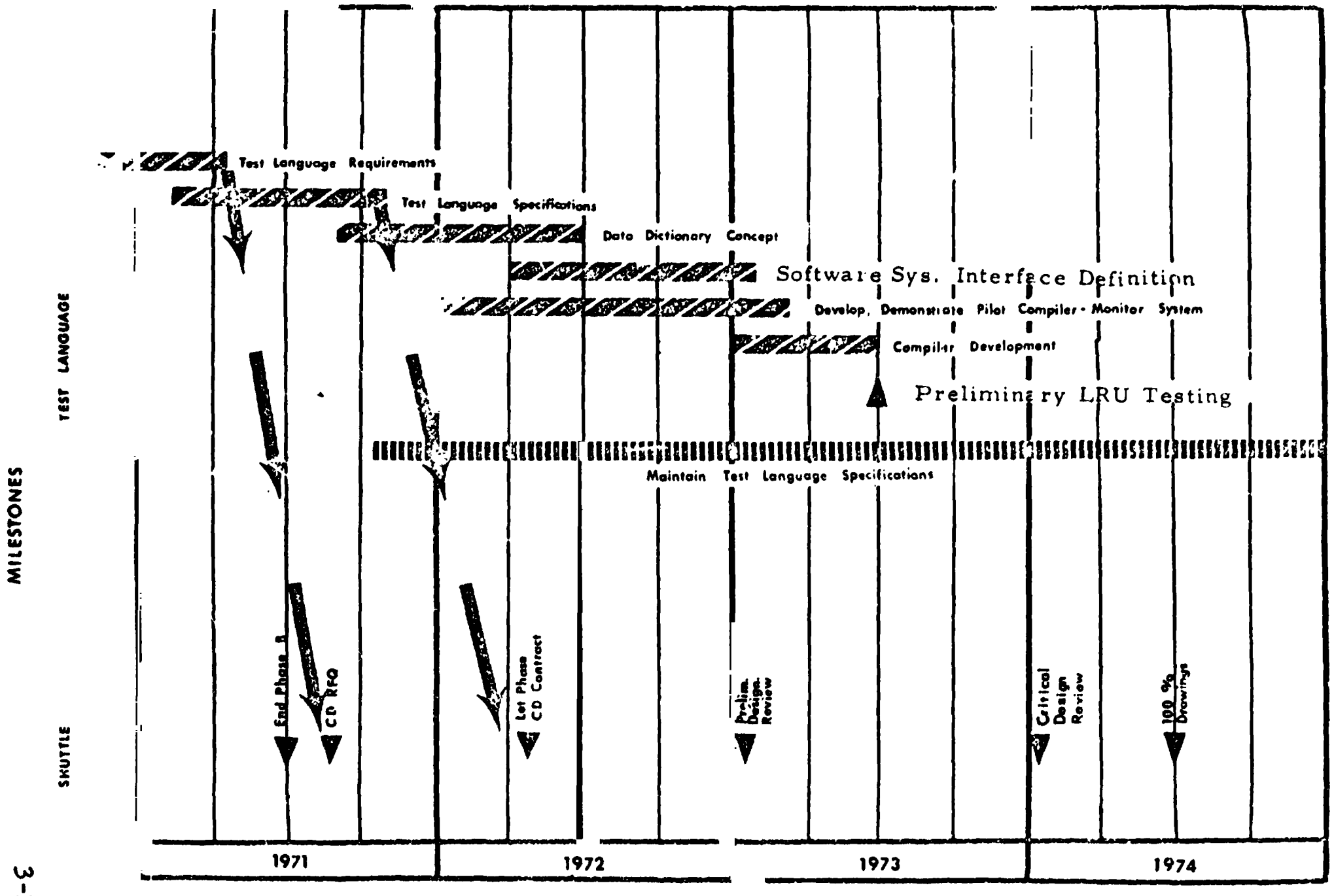

Figure 1. TEST LANGUAGE DEVELOPMENT SCHEDULE

system, language executive/interpreter, guidance and navigation, and the test and monitor capability using the language. Once decisions have been made in this area, the compiler development for generating the program for onboard computers can proceed. It is visualized that a parallel effort will be undertaken for off-board Line Replaceable Unit testing.

3.6 COMPILER DEVELOPMENT

The Compiler Development can be undertaken once the Language Specifications, Data Dictionary, and Software System Interface have been defined. It is anticipated that Line Replaceable Unit Testing, using the language, could be performed at an early date.

REFERENCES

1. Development of a Test and Flight Engineering Oriented Language, Phase III Report, Martin Marietta Corporation, NASA-KSC, MCR-70-424

2. Development of a Test and Flight Engineer Oriented Language, Final Report, Volumes I and II, M & S Computing, Inc., NASA-KSC, Report Number 70-0034

3. Development of a Test and Flight Engineering Oriented Language, Phase II Report, Martin Marietta Corporation, NASA-KSC, MCR-70-365

4. Development of a Test and Flight Engineer Oriented Language, Phase II, M & S Computing, Inc., NASA-KSC, Report Number 70-0031

5. Development of a Test and Flight Engineer Oriented Computer Language, Technical Proposal, M & S Computing, Inc., NASA-KSC, Reference RFP 3-309-0

APPROVAL

KSC-TR-1111

Requirements for a Standard Language for Test and Ground Operations

Originator:

Joe R. Medlock, Deputy Chief
Checkout Automation & Programming Office

Concurrence:

Walter J. Kapryan, Director of
Launch Operations

NASA-KSC MAY/71